TRACE FACILITY FOR USE IN MULTIPROCESSING ENVIRONMENT

BACKGROUND OF THE INVENTION

The development of software in computer based systems has become very time-consuming and, therefore, an expensive part of the initial cost of these systems. One difficulty is that many of the transactions between various pieces of software are not fully tested at the time the software is written. Accordingly, running the software causes logic errors to occur that are unexpected and these errors frequently produce undesirable results. These unexpected logic errors and undesirable results are frequently said to be due to bugs in the software.

One technique for locating the bugs in software is to intersperse coded statements within the software that provide the developer with an indication as to what values key parameters have during the execution of the software at the various points in the software at which the coded statements are inserted. The parameter values provided in this way will frequently provide the developer with an indication as to what is wrong with the present state of the software. These debugging statements frequently referred to as tracing statements take time to execute, and if a large number of them are interspersed within the code, the execution time for a particular piece of software can be increased significantly. It is therefore desirable to have only those tracing statements which are essential to a particular debugging operation to be executed. Inasmuch as there is usually no foresight as to which part of the software code or which function within the code will experience difficulties in execution, the tracing statements are usually interspersed at all points in a process that will be expected to give useful output parameters. This is found to be far more efficient than subsequently inserting statements only in the portions of code that are presently experiencing difficulties. The latter procedure would require constant rewriting of the source code and recompilation of that code every time a new bug is uncovered.

With trace statements interspersed throughout the code, it is desirable to have some way of deactivating the trace statements that are not presently required. One technique of achieving this mode of operation is to cause each tracing statement to contain arguments which indicate the subsystem and function with which it is associated in the code and to have a table loaded at the beginning of the process execution which specifies the subsystems and functions within the code presently requiring tracing. The inclusion of a subsystem or function within the table will cause the trace statements associated with those subsystems and functions to produce an output of the parameters specified in the trace statement. One difficulty with this approach is that the table must be changed in order to change the subsystem or function under investigation and the process must then be rerun from its starting point. This approach is especially difficult in cases where the processes execute over long periods of time. For example, in a communication system where there are many interrelated processes that are executed and continue to live while the system is active, a change in the table would require that the entire system be deactivated while a change be made in the table of each process. The system would then have to be reinitialized by bringing up all the processes and thereby consuming a large amount of time solely for the purpose of debugging a new subsystem or function.

Still another approach to the selective activation of trace statements in a multiprocessing environment is to provide a global table rather than a local table within each process. This global table can be associated with an operating system which is present during the execution of every process in the system. One feature that may be used to construct a table in connection with the VAX11/780 computer from Digital Equipment Corporation is the "logical name table" associated with its VMS operating system. Each process during its execution may then check the subsystem name and function name within the logical name table in order to determine whether a particular trace statement should be executed. The difficulty with this approach is that accessing the logical name table in the operating system is far more time consuming than accessing a table within the process itself. In addition, the logical name table may frequently have a much larger number of entries than a local table within the process and for this reason requires even more time than access to a local table.

SUMMARY OF THE INVENTION

A highly efficient trace facility is provided in accordance with the present invention wherein a process in a multiprocessing environment has a trace library within which a local trace table provides a list of the subsystems and functions whose trace statements are to be activated during any execution of the process. A global trace table is also provided containing a list of the processes, subsystems and functions whose trace statements are to produce results during execution. The trace library also has a local sync word which can be compared to a global sync word in order to determine whether any changes had been made in the global table that have not yet been updated within the local table. Each call to the trace library of the process causes such a comparison to be made between the two sync words. If no difference exists the local table is used to determine whether that particular call to the trace library should result in the output of the parameters associated with that call. The parameters in the local table are therefore replaced by their values in the global table only when a difference is detected between the local sync word and the global sync word. As a result, the much longer period of time required for reading the global trace table is avoided except when changes are made in the global table.

It is a feature of the present invention that the global trace table and the local table in the trace library also include a level value. This level value permits different degrees of investigation into the operation of each process during development of the code. Each trace statement is assigned a value to indicate the degree of importance of that trace statement in the debugging of that process. Level 1 statements, for example, could be reserved for the trace statements that are essential in determining an overview of the operation of that process (for example, inputs to and outputs from the process). At the other extreme, a higher number level, level four for example, can be assigned to those trace statements which provide detailed information about the operation of the subroutines within the process. As a result, the trace facility can be caused to permeate the execution of the processes to varying degrees depending on the level